COS126/128: General Computer Science	Spring 1996

Problem Set # 8

Handed out: April 16, 1996 *Due: April 24, 1996*

This assignment introduces you to *computational biology*, a new science that uses computational techniques to do biological research. In terms of programming skills, you will learn how to pass command-line arguments to a C program. You will also learn about *hashing*, a trick used often to make programs efficient.

You will write a program that finds the genetic encoding of amino acids. The input to the program is a protein and a genetic sequence known to contain that protein. These terms are explained below.

Background:

For the purposes of this assignment, we simplify the biology a little bit. A *genetic sequence* is a string made up of one of the four characters `'a'`, `'c'`, `'g'`, or `'t'`. An *amino acid encoding* is a genetic sequence of length 3. Note that there are $4 * 4 * 4 = 64$ different amino acid encodings. There are only 25 amino acids, which we will refer to with the capital letters `'A'` through `'Y'`. Each of the 64 different amino acid encodings corresponds to at most one amino acid, but an amino acid might have more than one encoding. A *protein* is a sequence of amino acids.

Now, given a protein and a long genetic sequence known to contain that protein, we want to establish the genetic encodings of the amino acids in the protein. An example will illustrate the process that we use to do this. Suppose that we know that the protein `M S I Q H M R` is found somewhere in the genetic sequence `attgctagcaatgctagcaattgctagcaattcat` but we don't know where, and we don't know any encodings for the amino acids. The problem seems hopeless at first, but it turns out that a straightforward approach — called the *one-pass trial and error method* — works if the protein sequence is long enough.

First, try aligning the first characters:

```
attgctagcaatgctagcaattgctagcaattcat
M  S  I  Q  H  M  R
```

This would work if a code for `M` were `att`, a code for `S` were `gct`, etc. But this can't be, because `gct` cannot be an encoding for both `S` and `H`. Note that having two different genetic encodings for an amino acid (`M` in this case) is allowed—what is required that each genetic code represent only one amino acid.

The sequences don't align at the beginning, so we next try aligning the protein sequence with the second character of the genetic sequence:

```
attgctagcaatgctagcaattgctagcaattcat
 M  S  I  Q  H  M  R
```

Again, `cta` can't encode two different amino acids, so this can't be the alignment. The same thing happens again for aligning the protein with the third character of the genetic sequence; how about the fourth character?

```
attgctagcaatgctagcaattgctagcaattcat
   M  S  I  Q  H  M  R
```

Not only is there a conflict with `agc` encoding both `S` and `H`, but also `gct` encodes both `M` and `Q`. Continuing to slide the protein sequence along the genetic sequence in this way, we finally get an alignment that could work:

```
attgctagcaatgctagcaattgctagcaattcat
          M  S  I  Q  H  M  R
```

This alignment is evidence that `atg` and `agc` are genetic codes for the amino acid `M`, `cta` is a code for `S`, `gca` for `I`, etc.

This is an artificial example to illustrate how the method works. In real data, the sequences are much much longer, and there's no mistaking a match. Furthermore, errors never happen: the same amino acid encodings always come out any time the method is run on a genetic sequence known to contain a protein. (These are also the encodings that you can find in any biochemistry textbook.)

Discovering the the relationships between genes and amino acids took several years of experimental work. Computational tricks like the one above are only now coming into widespread use. One wonders if the use of computers could have eliminated several years of difficult experimental work. In any case, this "string processing" abstraction (and other computational methods) for studying genetic material will certainly play an important role in biology in future.

The assignment:

You may start with the following program, found in `/u/cs126/progs/genecodes.c`. This uses UNIX file input to take filenames as arguments and read in the two needed sequences.

```
#include <stdio.h>
#include <stdlib.h>
int hash(char *p);
void unhash(int);  /* prints out 3 characters */
main(int argc, char *argv[])
  {
    char geneseq[10000], protseq[1000];
    char genecode[64];
    int i, j, c; char *p;
    FILE *prot = fopen(*++argv, "r");
    FILE *gene = fopen(*++argv, "r");
    if ( prot == NULL || gene == NULL ) exit(1);
    for (i = 0; (c = getc(prot)) != EOF; )
      if (c >= 'A' && c <= 'Z') protseq[i++] = c;
    protseq[i] = '\0';
    for (i = 0; (c = getc(gene)) != EOF; )
      if (c >= 'a' && c <= 'z') geneseq[i++] = c;
```

```
    geneseq[i] = '\0';

    /* your code here: print out where match is found */

    /* this prints out the encodings  */
    /* if a genetic sequence encodes no amino acid, print a blank */
    for (j = 0; j < 64; j++)
       { unhash(j);
         printf(" %c    ", genecode[j]);
         if (!((j+1) % 4)) printf("\n");
        }
  }
```

You will need to implement the alignment procedure in `main` and two auxiliary procedures: `hash`, which produces a different six bit number for each three-character genetic encoding, and `unhash`, which prints out the three characters corresponding to each six bit number. This allows you to use the table `genecode` (of size 64) in the alignment process to keep track of the encoding assignments that have been made. (Note that, in general, a "hash function" refers to a function that maps strings of characters to integers.) Your code must interface properly with what we have done in the program fragment that we are giving you above.

The files `/u/cs126/data/prot.1` and `gene.1` contain the sample data above for debugging. The files `prot.2` and `gene.2` in that directory contain real genetic data: the amino acid sequence for a "beta-lactamase" protein and the genetic sequence for a human virus.

What to submit: You should submit your program as "genecodes.c". For your hardcopy submission, include your output for the input files `prot.2` and `gene.2`.

Due: Wednesday, April 24

Submit electronically (`/u/cs126/submit`) before 10:00PM on Wednesday, April 24. Annotated hardcopy must be handed in during lecture on Thursday, April 25.